Analysis of Derivative-Free Direct Search Algorithms

An assignment for CSCI6514 Search and Optimization Course

Shakti Singh
Dalhousie University
Sh616913@dal.ca

ABSTRACT

This paper provides an analysis of Direct search algorithms on various unconstrained optimization problems. The algorithms chosen for this analysis are the Nelder-Mead method in the form of its MATLAB implementation *fminsearch* and Pattern Search method as its MATLAB implementation *patternsearch*. Different test problems like Sphere function, Noisy Sphere, Rosenbrock and Ellipsoid functions were used for the analysis. It was noted that *fminsearch* generally works better in lower dimensions of given unconstrained problems. *patternsearch* on the other hand works faster while fails to converge in Rotated Ellipsoid functions.

ACM Reference Format:
Shakti Singh. 2018. Analysis of Derivative-Free Direct Search Algorithms: An assignment for CSCI6514 Search and Optimization Course. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

Optimization of a function is the process of searching in the parameters space, the ones at which function value is maximum or minimum. It has vast applications throughout mathematics and science. Several algorithms have been developed to perform this search. Some make use of the derivatives or partial derivatives, others don't. Our focus in this paper will be on derivative-free algorithms also called as Direct Search method. Derivative-free algorithms use other metrics like Simplex and

0.1 Nelder-Mead Algorithm

The Nelder-Mead method also called as Simplex method is a Direct search algorithm that is used to find the minimum and maximum of an objective function. This method is a derivative-free method that uses Simplex to perform optimization steps. A Simplex is the convex hull of n+1 points in n-dimensional space. A Simplex maintains a set of n+1 points in an n-dimensional space. The algorithm interpolates between these points, observing the behavior of the objective function and then choosing a new point from them. The working of the Nelder-Mead algorithm is described as follows:

(1) In every iteration, the n+1 vertices are ordered in descending order of their corresponding function values.
(2) The centroid is computed for every point but the point having worst function value.
(3) To replace the worst vertex, Reflection along with Expansion, Contraction of the simplex is done to find a better point. A point having lower function value.
(4) If the above-mentioned steps failed to find a new point, the simplex s shrunk and the whole process is repeated again with new vertices.

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.475/123_4

0.2 Pattern Search Algorithm

Pattern Search Algorithm for optimization belongs a subclass of derivative-free algorithms in which the iterator looks in certain directions for a lower function value point. The directions for search is estimated either using Pattern Search or Coordinate Search. Pattern Search iterator chooses a set of directions around the current point. These directions are spread out around the current point like a frame, giving an insight into the curvature. Function values are calculated along these directions and the current point is shifted to the point having the lowest function value. The process repeats if successful, otherwise new directions are added and the step size is increased. A key condition is that one of the direction from the set of directions should give a descent direction if the current point is not a stationary point.

1 MOTIVATION

This paper sheds light on the performance of Direct search algorithms on various optimization problems. The behavior of algorithms differs from one problem to other. The effect of introducing noise and ill-conditioning in optimization problems will also be analyzed.

2 EXPERIMENTS

The experiments for this paper were conducted in MATLAB software and the code for reproducing the results have been provided in the codes repository. These experiments include execution of *fminsearch* method of MATLAB as an implementation of the Nelder-Mead strategy and *patternsearch* as an implementation of Pattern Search. These methods are tested on Sphere function, RosenBrock function, Noisy Sphere function, Cigar function, and ill-conditioned Cigar function. The results of these experiments are reported in the form of convergence graphs. Every problem will be tested on multiple dimensional inputs. To reproduce the results of the experiments, execute fminsearchtests.m and pattersearchtests.m form the code directory.

2.1 Sphere function

Sphere function is a simple optimization objective function that is of the shape of a Convex bowl with the minimum at the Origin. It is defined as

$$y = x^T * x \tag{1}$$

where x can be of multiple dimensions.
The parameters of the algorithms were left as default for the initial tests where Maximum iterations are 200 times the number of variables, Similar value holds for Maximum function evaluations. Termination tolerance on function value and input is set to 1e-4. These parameters are kept constant for all the problems tested on *fminsearch*.
Test results for *patternsearch* on algorithms are also depicted below in the form of optimization graphs. The options of *patternsearch* are left as default values to check the performance without tweaking. Test results of both the optimization strategies are depicted below as convergence plots.

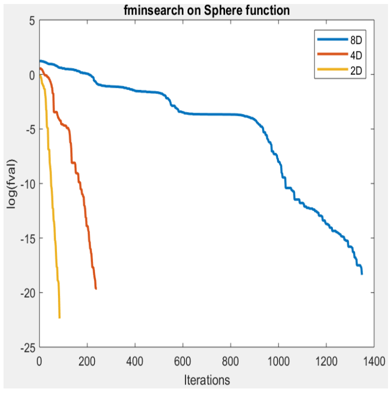

Figure 1: Performance of *fminsearch* on Sphere function

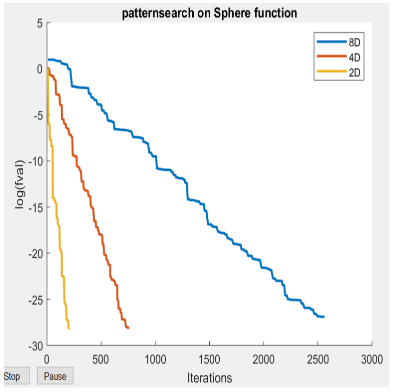

Figure 2: Performance of *patternsearch* on Sphere function

The tests were run for several dimensions. Both of the strategies worked considerably well when using lower-dimensional inputs, while performance starts to affect when increasing the dimensions. Next, we will examine the performance on Noise-induced Sphere function.

2.2 Noisy Sphere function

A certain degree of Noise is induced in the sphere function to test the ruggedness of the algorithms. The new equation can be written as

$$y = x^T * x * (1 + v * N) \tag{2}$$

where v is the noise strength parameter while N is a normally distributed variable. The strength is kept constant to 1e-6 for all the tests.
The performance of the strategies on the Noisy sphere of various dimensions is depicted and explained below.

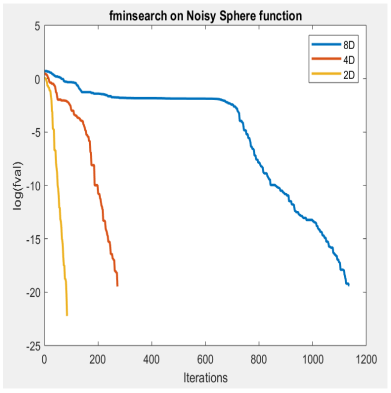

Figure 3: Performance of *fminsearch* on Noise induced Sphere function

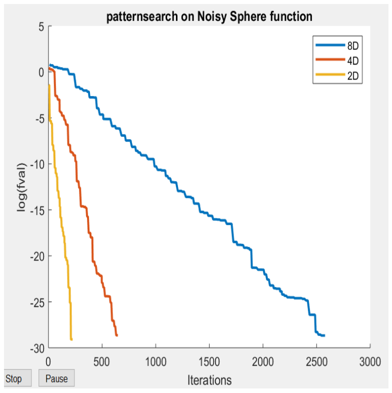

Figure 4: Performance of *patternsearch* on Noise induced Sphere function

2.3 Rosenbrock Function

Rosenbrock function also called as Banana optimization function is generally used to test the algorithm's ability to follow a curved valley. The function can be formulated as:

$$y = 100 * (x(2) - x(1)^2)^2 + (1 - x(1))^2; \tag{3}$$

the local minima of this function lies in the curved valley at f(1,1...1) = 0.

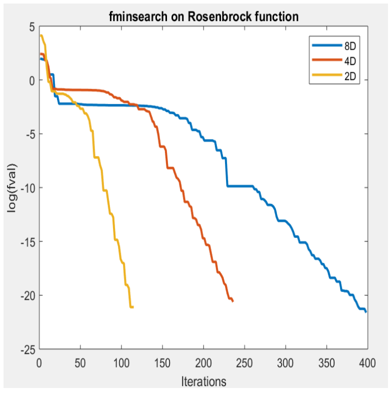

Figure 5: Performance of *fminsearch* on Rosenbrock function

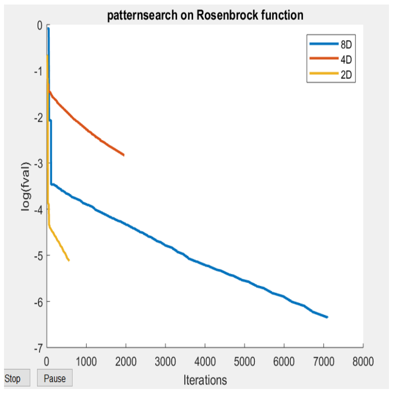

Figure 6: Performance of *patternsearch* on Rosenbrock function

The *fminsearch* strategy worked considerably well on Rosenbrock function in all the tests for different dimensions. On the other hand, convergence totally fails while using *patternsearch* on Rosenbrock. This behavior can be accounted to the inefficiency to find a direction towards minimum when the iterator reaches a saddle point or a Valley. The performance did not improve even on increasing the number of iterations or function evaluations.

2.4 Cigar Function

Cigar function belongs to a new family of optimization functions called Ellipsoid test functions. This class of functions contains convex quadratic functions of the form

$$y = x^T * A * x \tag{4}$$

These type of functions are used to test the performance of the convergence algorithms on ill-conditioned problems. The rate of convergence of the algorithm is inversely proportional to the condition number of the Hessian matrix calculated. For Cigar function the matrix A will take the form

$$A = diag(\alpha, 1, ..., 1), \alpha << 1 \tag{5}$$

If alpha « 1, the function acts as a Cigar function while if alpha » 1 then it acts a Discus function.

The performance of the algorithms on Cigar function in different dimensions is shown and discussed as follows:

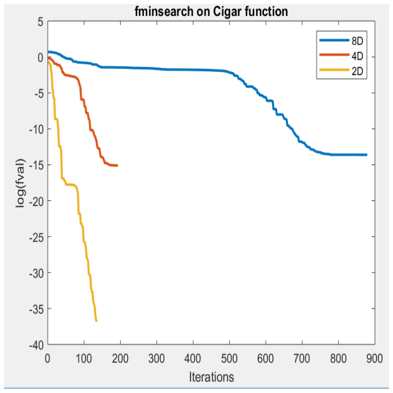

Figure 7: Performance of *fminsearch* on Cigar function

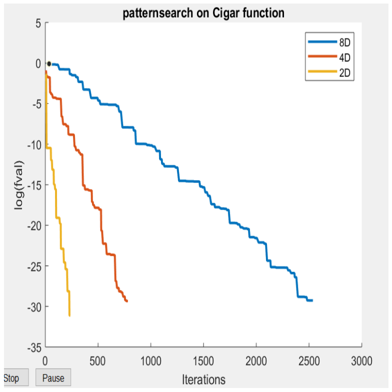

Figure 8: Performance of *patternsearch* on Cigar function

The *fminsearch* strategy works well on Cigar function in lower dimension while as we increase the dimensions, the performance starts to degrade. *patternsearch* algorithm works great even on higher dimensions of the Cigar functions. The failure of *fminsearch* in higher dimensions can be because of it getting stuck on local minimum or a saddle point.

2.5 Rotated-Cigar function

Rotated version of the Ellipsoid functions is used to restrain the algorithm to avoid exploitation of the separability to find the next point.

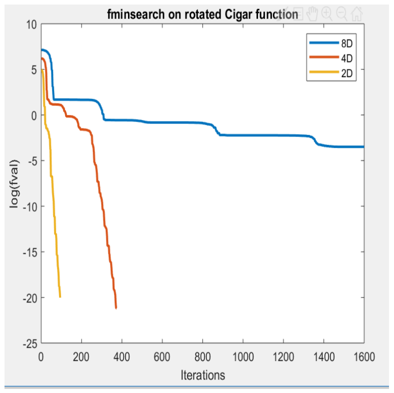

Figure 9: Performance of *fminsearch* on Rotated Cigar function

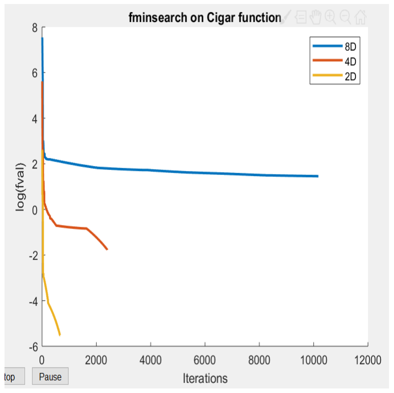

Figure 10: Performance of *patternsearch* on Rotated Cigar function

The performance of *fminsearch* gets better as we use it on the rotated Cigar function. It successfully converges to a non-stationary point in 2D and 4D input while still dabbles in 8D and higher input dimensions. *patternsearch* on the other hand fails to converge to any non-stationary point in Rotated Cigar. This can be because of the algorithm exploiting separability in Cigar, thus good performance in it, while it wasn't able to do so in the case of Rotated- Cigar function.

3 CONCLUSION

- The Sphere function turned out to be very simple optimize for both of *fminsearch* and *patternsearch*. Both of the algorithms converged to a point very close to the global minimum. Convergence pattern observed was almost linear(y-axis is on a logarithmic scale) in the case of *patternsearch* and it worked faster than *fminsearch*.
- The performance on the Noisy sphere was also similar to that on Non-noisy one. The amount of noise-induced does not seem to affect the performance in any considerable manner.
- Rosenbrock function poses a problem for the *patternsearch* strategy. The algorithm failed to converge anywhere near the minimum. While the *fminsearch* algorithm converged successfully to a non-stationary point near the minimum. A general pattern of sudden decrease and then a gradual decrease is observed when higher dimensional input was utilized.
- The performance on Ellipsoid function was also a mixed affair for both of the algorithms. Both *fminsearch* and *patternsearch* converged to a non-stationary point in the case of Cigar function. Although, the performance started to degrade when dimensions were increased. The *fminsearch* seemed to be getting stuck in the valley around 10e-10 log(fval) from which it manages to get out in lower dimension but not in higher dimensions.
- The problem of Rotated Cigar function proved out to be challenging for both the methods. The *fminsearch* worked comparatively well to *patternsearch* as it manages to converge near the global optimum in a lower dimensional input. The *patternsearch* failed to converge.

4 FUTURE SCOPE OF WORK

There are several directions that can serve the purpose of future work in this paper:

- Trying out various configurations of the underlying algorithms to better analyze the effect of every parameter.
- Different problems of convex, non-convex, quadratic or non-quadratic nature can be used to test the performance of algorithms
- Netter visualization techniques like plotting the Mesh-grid of the underlying problems and then plotting the iterate will help to get better answers related to the behavior.

5 REFERENCES

(1) Jorge Nocedal and Stephen J. Wright. 2006. Numerical optimization New York: Springer.
(2) Algorithm implementations https://www.mathworks.com/matlabcentral/fileexchange/23147-test-functions-for-global-optimization-a
(3) Banana function optimization https://www.mathworks.com/help/optim/examples/banana-function-minimization.html
(4) Patternsearch documention https://www.mathworks.com/help/gads/patternsearch.html?s_tid=doc_ta
(5) fminsearch documention https://www.mathworks.com/help/matlab/ref/fminsearch.html